```
// A Finger Gazing BEEST on Table
// Copyright (C) 2017 ArduinoDeXXX All Rights Reserved.
//
// Materials:
// "Arduino NANO",
// A motor driver IC "L298N"
// 3 Infrared (IR) phototransistors ("OP505A", 3pcs) and 3 resistors (10k ohm)
// 3 IR LEDs ("SFH4554DWEW", 3pcs) and 3 resistors (100 ohm)
// A white light LED and a resistor (100-1000 ohm) ... They can be omitted.
// (If they are omitted, #13 pin should be substituted for #8 pin in this sketch.)
//
// Information:
// 1S LiPo battery (3.7V) can be used for 2 DC motors. Connect its positive to pin4 on L298N.
// It supplies power for 3 IR LEDs.
// 6P battery (9V) can be used for Arduino NANO. Connect its positive to VIN-pin on NANO.
//
// How to command:
// Shaking finger at a distance from an eye: BEEST steps forward
// Moving finger or palm near to an eye: BEEST steps backward
// Keeping finger far from every three eye: BEEST keeps still at rest
// Moving finger or palm near to an eye quickly twice in a row: BEEST stops its stepping
//
// View the sites bellow to see more detail.
// http://www.instructables.com/id/Training-Theo-Jansens-Mini-BEEST/
// http://www.instructables.com/id/Training-Theo-Jansens-Mini-BEEST-JPN/


int row = 0;

long R[16];  // Outputs of phototransistor
long C[16];
long L[16];

long R0 = 0;  // The normal value of output above
long C0 = 0;
long L0 = 0;

long dR[25];  // Change of output with 15 lags
long dC[25];
long dL[25];

long sumDr = 0;  // Sum of elements dR[25]
long sumDc = 0;
long sumDl = 0;

long sumSqDr = 0;  // Sum square of elements dR[25]
long sumSqDc = 0;
long sumSqDl = 0;

long devDr = 0;  // Standard deviation of elements dR[25]
long devDc = 0;
long devDl = 0;

boolean checkLumi = false;
int checkMode = 0;
long recTime = 0;

const int checkDev = 55;
const int goUniform = 500;

int countDev = 0;
```

```
const int callDev = 8;
const int goForward = 70;
const int goBack = 150;



//******************************************************************************

void setup() {
  pinMode(11, OUTPUT); // M1
  pinMode(12, OUTPUT);
  pinMode(5, OUTPUT); // PWM1
  pinMode(6, OUTPUT);  // M2
  pinMode(7, OUTPUT);
  pinMode(3, OUTPUT); // PWM2

  pinMode(8, OUTPUT); // LED for communication (If onbord LED used, replace every #8 with #13 in this sketch.)

  delay(500);

  // Select value of PWM
  digitalWrite( 8 , HIGH );  delay(400);  // First, blink LED 3 times.
  digitalWrite( 8 , LOW );   delay(700);
  digitalWrite( 8 , HIGH );  delay(400);
  digitalWrite( 8 , LOW );   delay(700);
  digitalWrite( 8 , HIGH );  delay(400);
  digitalWrite( 8 , LOW );   delay(2000);
  R[0] = analogRead(A0);
  C[0] = analogRead(A1);
  L[0] = analogRead(A2);
  long maxLumi = max( R[0] , max( C[0] , L[0] ) );
  if ( R[0] == maxLumi ) {  // Select High speed
    analogWrite( 5, 255 );
    analogWrite( 3, 255 );
    digitalWrite( 8 , HIGH );   delay(500);  // Blink LED 3 times.
    digitalWrite( 8 , LOW );    delay(400);
    digitalWrite( 8 , HIGH );   delay(500);
    digitalWrite( 8 , LOW );    delay(400);
    digitalWrite( 8 , HIGH );   delay(500);
    digitalWrite( 8 , LOW );    delay(2000);
  }
  else if ( C[0] == maxLumi ) {  // Sellect middle speed
    analogWrite( 5, 215 );
    analogWrite( 3, 215 );
    digitalWrite( 8 , HIGH );   delay(800);  // Blink LED twice.
    digitalWrite( 8 , LOW );    delay(400);
    digitalWrite( 8 , HIGH );   delay(800);
    digitalWrite( 8 , LOW );    delay(2000);
  }
  else {  // select low speed
    analogWrite( 5, 170 );
    analogWrite( 3, 170 );
    digitalWrite( 8 , HIGH );   delay(1000);  // Blink LED only once.
    digitalWrite( 8 , LOW );    delay(2000);
  }

  // Getting the normal value of output of each phototransistors
  uniform();

  // Getting initial values
  for ( int i=15 ; i>=0 ; i-- ) {
    R[i] = analogRead(A0) - R0;
    C[i] = analogRead(A1) - C0;
    L[i] = analogRead(A2) - L0;
    delay(5);
  }

  for ( int k=24 ; k>=0 ; k-- ) {
    for ( int j=15 ; j>0 ; j-- ) {
      R[j] = R[j-1];
      C[j] = C[j-1];
      L[j] = L[j-1];
    }
    R[0] = analogRead(A0) - R0;
    C[0] = analogRead(A1) - C0;
    L[0] = analogRead(A2) - L0;
    dR[k] = R[0] - R[15];
    dC[k] = C[0] - C[15];
    dL[k] = L[0] - L[15];
    sumDr = sumDr + dR[k];
    sumDc = sumDc + dC[k];
    sumDl = sumDl + dL[k];
    sumSqDr = sumSqDr + sq( dR[k] );
    sumSqDc = sumSqDc + sq( dC[k] );
    sumSqDl = sumSqDl + sq( dL[k] );
    delay(4);
  }
}

//**************************************************************************

void loop() {

  sumDr = sumDr - dR[24];
  sumDc = sumDc - dC[24];
  sumDl = sumDl - dL[24];

  sumSqDr = sumSqDr - sq( dR[24] );
  sumSqDc = sumSqDc - sq( dC[24] );
  sumSqDl = sumSqDl - sq( dL[24] );

  for ( int i=15 ; i>=1 ; i-- ) {
      R[i] = R[i-1];
      C[i] = C[i-1];
      L[i] = L[i-1];
  }
  R[0] = analogRead(A0) - R0;
  C[0] = analogRead(A1) - C0;
  L[0] = analogRead(A2) - L0;

  for ( int k=24 ; k>=1 ; k-- ) {
    dR[k] = dR[k-1];
    dC[k] = dC[k-1];
    dL[k] = dL[k-1];
  }
  dR[0] = R[0] - R[15];
  dC[0] = C[0] - C[15];
  dL[0] = L[0] - L[15];

  sumDr = sumDr + dR[0];
  sumDc = sumDc + dC[0];
```

```
sumDl = sumDl + dL[0];

sumSqDr = sumSqDr + sq( dR[0] );
sumSqDc = sumSqDc + sq( dC[0] );
sumSqDl = sumSqDl + sq( dL[0] );

devDr = sqrt( sumSqDr / 25  - sq( sumDr / 25 ) );
devDc = sqrt( sumSqDc / 25  - sq( sumDc / 25 ) );
devDl = sqrt( sumSqDl / 25  - sq( sumDl / 25 ) );

// The maximum of brightness
long maxLumi = max( R[0] , max( C[0] , L[0] ) );

// The maximum of variation of brightness
long maxDev = max( devDr , max( devDc , devDl ) );

// Average brightness except for the maximum
long lowLumi = ( R[0] + C[0] + L[0] - maxLumi ) / 2;

// Average variation except for the maximum
long lowDev = ( devDr + devDc + devDl - maxDev ) / 2;


if ( maxDev - lowDev > checkDev ) {
  if ( checkMode == 2 && millis() - recTime < goUniform ) {
    uniform();
    checkMode = 0;
  } else {
    checkMode = 1;
    recTime = millis();
  }
} else {
  if ( checkMode == 1 ) {
    checkMode = 2;
  } else {
    if ( checkMode !=2 || millis() - recTime > goUniform ) {
      checkMode = 0;
    }
  }
}



// Decision for stepping
if ( maxLumi - lowLumi > goBack ) {  // Backward
  checkLumi = false;
  countDev = 0;
  if ( R[0] == maxLumi ) {
    rightB();
  }
  else if ( C[0] == maxLumi ) {
    back();
  } else {
    leftB();
  }
}

else if ( maxDev - lowDev > callDev ) {  // Forward
  checkLumi = false;
  countDev++;
  if ( countDev > goForward ) {
    if ( devDr == maxDev ) {
        right();
      }
      else if ( devDc == maxDev ) {
        forward();
      } else {
        left();
      }
    }
  }

  else {  // wait
    countDev = 0;
    stopping();
  }

  delay(5);
}


//*************************************************************

void uniform() {
  digitalWrite ( 8 , HIGH );
  stopping();
  delay(700);
  digitalWrite ( 8 , LOW );
  delay(350);

  R0 = 0;
  C0 = 0;
  L0 = 0;
  for ( int i=0 ; i<=300 ; i++ ) {
    R0 = R0 + analogRead(A0);
    C0 = C0 + analogRead(A1);
    L0 = L0 + analogRead(A2);
  }
  R0 = R0 / 300;
  C0 = C0 / 300;
  L0 = L0 / 300;

  for ( int i=0 ; i<16 ; i++ ) {
    R[i] = 0;
    C[i] = 0;
    L[i] = 0;
  }
  for ( int k=0 ; k<26 ; k++ ) {
    dR[k] = 0;
    dC[k] = 0;
    dL[k] = 0;
  }
  sumDr = 0;
  sumDc = 0;
  sumDl = 0;

  sumSqDr = 0;
  sumSqDc = 0;
  sumSqDl = 0;

  digitalWrite ( 8 , HIGH );
  delay(350);
  digitalWrite ( 8 , LOW );
```

```
}

//************************************************************

void forward() {
  digitalWrite( 11, HIGH );
  digitalWrite( 12, LOW );
  digitalWrite( 6, LOW );
  digitalWrite( 7, HIGH );

}
void stopping() {
  digitalWrite( 11, LOW );
  digitalWrite( 12, LOW );
  digitalWrite( 6, LOW );
  digitalWrite( 7, LOW );
}
void back() {
  digitalWrite( 11, LOW );
  digitalWrite( 12, HIGH );
  digitalWrite( 6, HIGH );
  digitalWrite( 7, LOW );
}
void left() {
  digitalWrite( 11, HIGH );
  digitalWrite( 12, LOW );
  digitalWrite( 6, LOW );
  digitalWrite( 7, LOW );
}
void right() {
  digitalWrite( 11, LOW );
  digitalWrite( 12, LOW );
  digitalWrite( 6, LOW );
  digitalWrite( 7, HIGH );
}
void leftB() {
  digitalWrite( 11, LOW );
  digitalWrite( 12, HIGH );
  digitalWrite( 6, LOW );
  digitalWrite( 7, LOW );
}
void rightB() {
  digitalWrite( 11, LOW );
  digitalWrite( 12, LOW );
  digitalWrite( 6, HIGH );
  digitalWrite( 7, LOW );
}
```